# Indiana Music Business Directory: An Overview of a Web Development Project

**An Honors Thesis (CS 495/CS 498)**

by

Jason Toomey

**Thesis Advisor**

Lan Lin

Signed

**Ball State University**

**Muncie, Indiana**

May 2015

**Expected Date of Graduation**

May 2015

## Abstract

Web development is a complicated process that requires much technical knowledge. It is often difficult for non-specialists to understand all that is involved in a web development project. In this paper, I will give an overview of the Indiana Music Business Directory project in a way that is understandable to those without experience in web development. This project started in September 2014 as a Computer Science capstone project at Ball State University. The project can be divided into three main tasks: requirements specification, database design, and application development. I will explain each of these in detail and discuss some of the problems my group encountered during the project.

## Acknowledgements

I would like to thank Kevin Charlton and Yufeng Xue for all the hours they spent working on the Indiana Music Business Directory project. Without their help, the project would not have been such an extraordinary success.

I would also like to thank our client Robert Willey for all of his help.

Finally, I would like to thank my advisor, Dr. Lan Lin, for reading my paper and for her guidance throughout the project.

## Note

For your convenience, every figure used in this paper is included on the disk attached to the inside cover of the binder. The capstone project's final report is also on the disk. This report provides technical information that is not included in this paper.

# 1 Introduction

During the fall semester of my senior year at Ball State University, I started work on the Indiana Music Business Directory project along with Kevin Charlton and Yufeng (Mark) Xue. This work was to be done for our Computer Science capstone project under the direction of our professor Lan Lin. The project began in September 2014 after we accepted the project from our client, Robert Willey, the director of the Music Media and Production Industry program at Ball State. Our goal was to create a web application to manage data on music businesses in Indiana.

When the project began, the Indiana Music Business Directory consisted of several spreadsheets containing business data that were available to the public. Businesses were grouped into twelve categories as shown in Figure 1, and each category had its own spreadsheet. A web form for each category, such as the one shown in Figure 2, could be filled out to add a business to one of the spreadsheets (Figure 3).

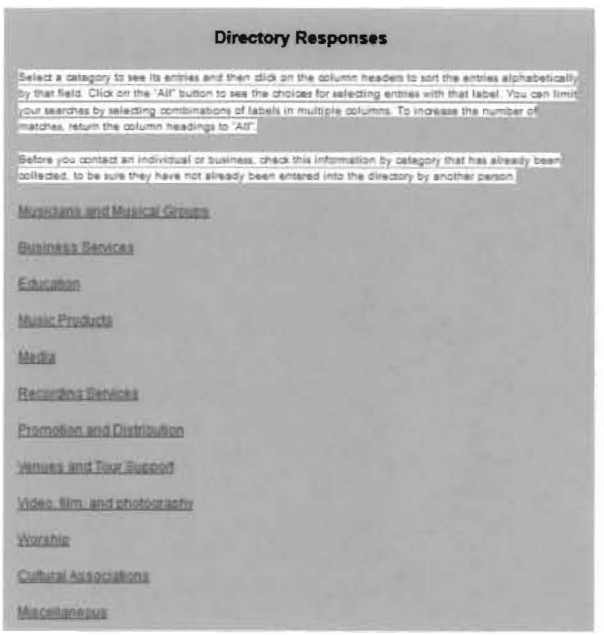


**Figure 1**. Twelve categories for businesses

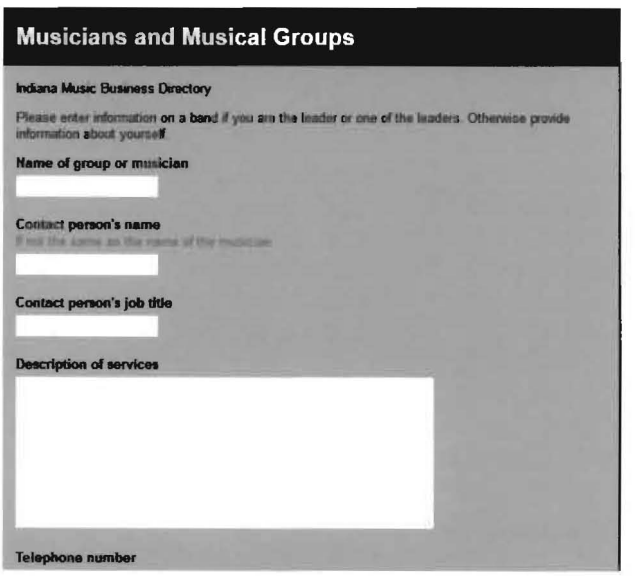


**Figure 2.** An example of a form used to add a business

| Company name | Column D | Contact person's name | Contact person's job title | Description of products and services | Phone number | Email address | Website(s) |
|---|---|---|---|---|---|---|---|
| (All) | (All) | (All) | (All) | (All) | (All) | (All) | (All) |
| Village Green Records | | Travis Harvey | Owner | Music of all styles and varieties, including new and vintage. Visit our music store, listen to your favorite tunes, scratch some vinyl records, and jam out. Travis is an excellent resource to help you find new music to suit your tastes, and to expand your horizons. | 847-346-2199 | villagegreenrecord | https://www.facebook.com /VillageGreenRecords http://villagegreenrecords.com |
| Muncie Music Center | | Sarah | Rental Accounts Manager | Everything to rentals to parts for instruments | 1-800-992-4481 | sales@munciemus | http://www.mmcrentals.com/ |
| Sight and Sound Music Center | | Ryan | Store Manager | Musical instrument and audio. Rental repair and lessons. | 1-765-289-8526 | retail@worldmusicc | http://www.sightandsoundmusic.com/ |
| Dan's Downtown Records | | Danial Walter | Owner/Manager /Worker | Sell music CDs, DVDs, Records, VHS, cassette tapes, etc.<br><br>They also do Cd duplicating, record transfers, and orders | (765)-284-7611 | NA<br><br>Check Facebook Page Dan's Downtown Records | http://www.dansdowntownrecords.com/ |
| World Music Supply | | Terry Renbarger | Shipping Manager | The store sells guitars and many different guitar accessories. They also sell drums as percussion instruments. The also sell sound systems | 1-765-213-6085 | sales@worldmusic | http://www.worldmusicsupply.com/ |
| IRC Music | | Jeff | sales | They offer many different brands of instruments as well as music lessons. They also deal in used, | (317)-849-7965 | ircmusic@sbcglob | http://ircmusicstores.com/ |

**Figure 3.** One of the twelve spreadsheets containing business data

### 1.1 Limitations

The old business directory had many limitations. First, it was difficult to search for a specific business because it could exist in one or more of the spreadsheets. Users would have to look through all the data in each spreadsheet to find the business they wanted. There was no way to filter out the businesses they did not want. Another limitation was difficulty maintaining the data. When a new business was added, the user was supposed to check if the business already existed. This was a difficult task especially since they would have to check all twelve spreadsheets to make sure the business was not already in the data. Duplicate and invalid data was common. Although the data could be updated by manually editing a spreadsheet, this was unlikely to happen and likely to introduce errors. Therefore, users could not be sure the data was accurate. Another limitation was that none of the data was validated before being added to the directory. For example, the data entry forms could not detect if a valid phone number was entered, and phone numbers were stored in many different formats. One final limitation was the inability to moderate who can modify the data. Since the spreadsheets were available to the public, anyone could change the data anonymously. Because of these limitations, a new system was needed.

### 1.2 Original Requirements

In the original project proposal, eight requirements were specified. These requirements are listed below:

- Users can search for businesses.
- Users can add businesses to the directory.
- Users can update business information if they have permission.
- Update permissions are linked to an account.

- Account passwords can be reset by sending an email.
- Music business students who manage the data can update any business.
- A web application using HTML must be built.
- At least one mobile application must be built for iOS or Android phones.

Many additional requirements were added as the project progressed. New features were developed from these requirements. An example is the review feature which allows users to rate a business and leave a comment. These new features will be discussed later.

### 1.3 Software Development Process

My group and I decided to follow a sequential approach to software development for this project. This approach is called the waterfall model because each step in the process is completed in succession. Since this project would be larger than any we had worked on before, we wanted to determine most of the requirements at the beginning before writing any code. This would help us understand the overall design of the system and give us a final goal we could reach. In his book, Pressman lists five steps in the software development process. These steps are communication, planning, modeling, construction, and deployment [1]. For our project, communication and planning were part of the requirements specification. Modeling included database and system design. Construction included building the database and web application. Deployment included uploading files to the web server. Since we followed the waterfall model, we first needed to complete the requirements specification before working on the design or writing any code.

## 2 Requirements Specification

Soon after my group accepted the project from Robert Willey, we scheduled a meeting to discuss the requirements in detail. As Willey talked about all the ideas he had for the project, I listened and considered how difficult each idea would be to implement. I asked questions whenever something was not clear. During this meeting, the original requirements were clarified and many new requirements were added. The new requirements included the following:

- Different types of accounts can be created depending on the user's privileges. These types include a normal account with limited privileges, an account for business owners, and an account for music business students who manage the data.
- Those with normal or student accounts can write reviews of the businesses. A review includes a comment and a rating from 1 to 5.
- Those with business or student accounts can add new businesses to the directory.
- Those with business accounts can update businesses they added, and those with student accounts can update any business.

- Moderator accounts can also be created. (optional requirement)
- Those with moderator accounts can flag reviews, delete reviews, suspend accounts, and delete businesses.

Since it was hard to find times to meet, we tried using videoconferencing for our meetings. I found these meetings to be unproductive and difficult. However, since our first milestone report was due soon, I convinced Robert Willey to meet with us in person. We discussed technical feasibility such as hosting for the website and which database management system to use. We found that the project was feasible and we could continue working.

After writing our first milestone report, we scheduled another meeting with Willey. At this meeting, we decided to change the account types and the privileges associated with them. We combined the privileges of the normal and business accounts from the old requirements into a single normal account. Now, those with a normal account could add businesses, write reviews, and update the businesses that they added. The business account type was no longer needed. A few additional requirements were discussed at this meeting as well. Now that we had the requirements, we could start designing the database.

# 3 Database Design

## 3.1 Data Analysis

There were three steps when designing the database: data analysis, data modeling, and schema design. First, we had to analyze the existing data from the old Indiana Music Business Directory. I did this myself and made several observations. Every category had some data fields in common. A single database table could be used to store business information that did not depend on a specific category. This included the business name and description. Another observation was that most listings in the directory included contact and location information. Some businesses existed in the directory more than once with different contacts or locations. Therefore, we needed database tables for contacts and locations. I also noticed that some businesses existed under more than one category. Because of this, it was necessary to separate category specific information from general business information.

## 3.2 Data Modeling

After data analysis, we needed to create a model for our database. Since we were using a relational database, we used the Entity-Relationship (ER) model represented in an ER diagram. We followed the process described in our database textbook to create the diagram and later to design the database schema [2]. To make the ER diagram easier to draw, we did not use ovals for the attributes but instead listed attributes next to their entity or relationship. The ER diagram is shown in Figure 4.

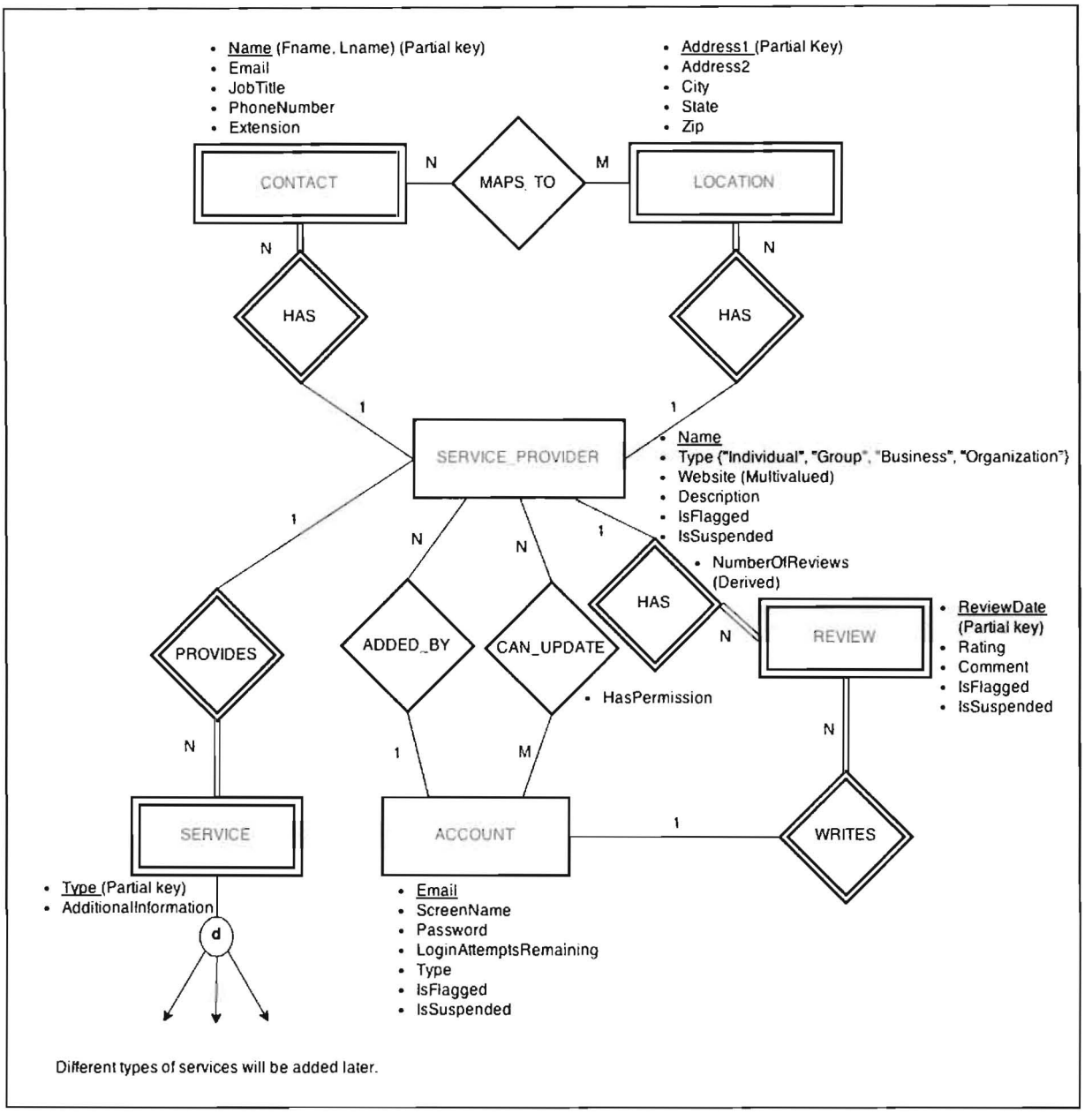


**Figure 4**. ER Diagram

There are several parts of this diagram that should be explained. First, we called the business entity a service provider because listings in the directory do not have to be businesses. Individual musicians, music groups, and organizations are also allowed in the directory. One important relationship is the *PROVIDES* relationship between the *SERVICE_PROVIDER* and *SERVICE* entities. Because a listing in the original directory can exist under more than one category, we needed to separate general business information from information specific to a category. We determined that the categories can be thought of as services and a service provider can provide any subset of services. For example, suppose a music store is listed under

the *Music Products* and *Education* categories because it sells guitars and offers guitar lessons. In this model, the music store is a service provider that provides two services: music product sales and education. The service provider entity contains the business information such as the store's name and a short description. The education table contains information specific to the service such as what instruments are taught and the music instructor's name. Similarly, the music products table contains information such as what products are sold.

Another important part of the diagram is the relationships between the *SERVICE_PROVIDER*, *CONTACT*, and *LOCATION* entities. Each service provider can have zero or more contacts and locations, but we needed a way to determine which contacts were for each location. For example, if the music store mentioned earlier has two locations, they might have a manager for each store. Each manager would be a separate contact linked to the business. However, users would need to know which manager to contact for one of the locations. The relationship between the *CONTACT* and *LOCATION* entities handles this problem.

### 3.3 Schema Design

The final step was to create the database schema which is the actual structure of the database. This included all tables and columns, referential constraints on the columns, and data types. The relational schema diagram is shown in Figure 5 on the next page. Although a few changes were made as the project progressed, the database structure closely matches what is shown in the diagram.

To understand this diagram, knowledge about relational databases is required. A relational database consists of one or more tables containing rows of data. Each row contains a value for every column in the table. A column has a name and a data type to give meaning to the values in the column. For example, a simple *CONTACT* table could contain the *Fname*, *Lname*, and *PhoneNumber* columns. A row in this table would be one person with a first name, last name, and phone number. Many contacts can be stored in this table, but all of the data would have the same structure.

Each table has a primary key used to uniquely identify a row in the table. In the simple CONTACT table just described, the *Fname* and *Lname* columns could be the primary key if you assume that no two contacts can have the same first and last names. Many more constraints can be placed on the columns such as requiring a non-null value, not allowing negative numbers, and requiring values to be unique.

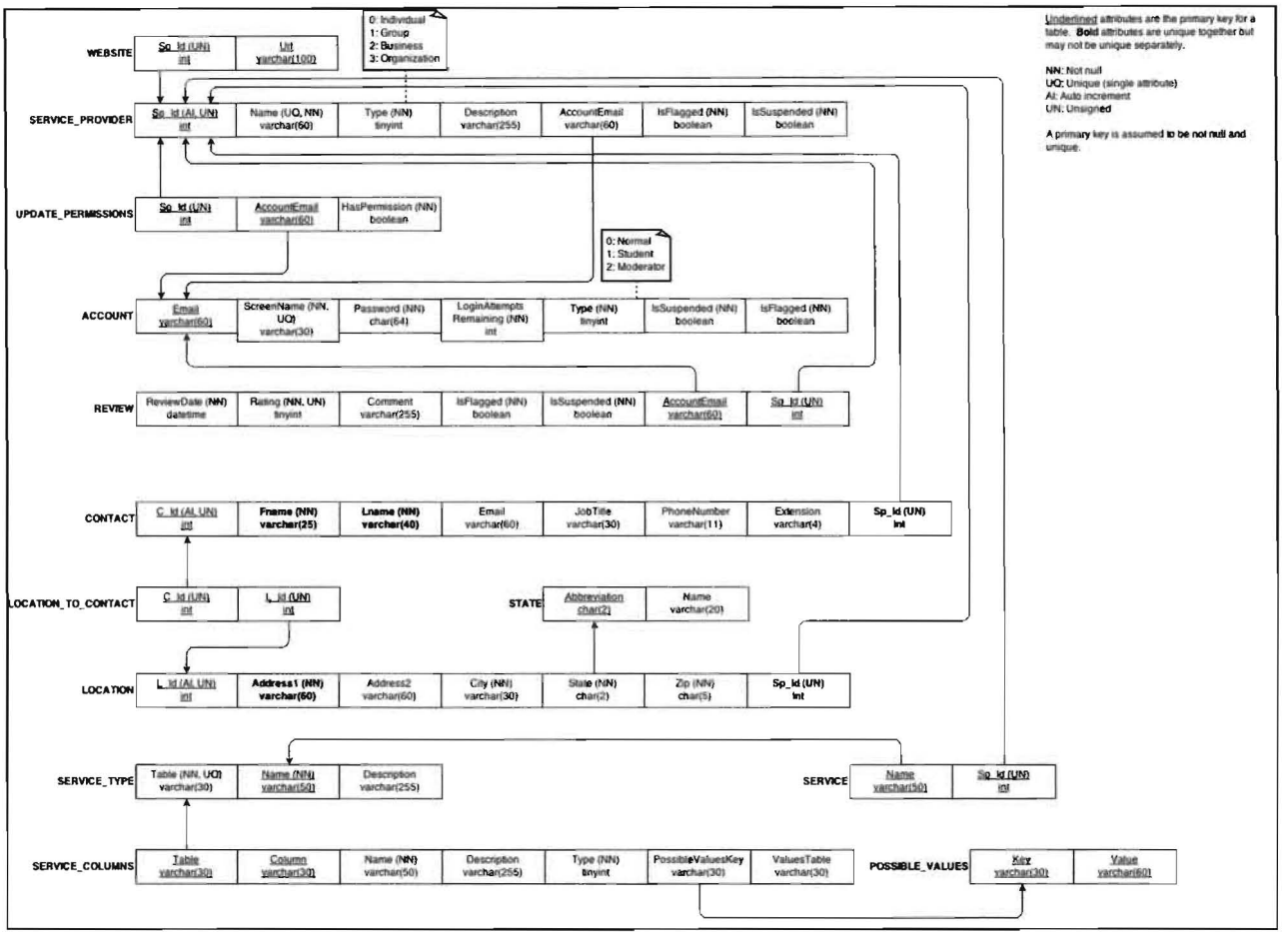


**Figure 5.** Relational Schema Diagram

In this diagram, the table names are listed in bold, to the left of two or more boxes. These boxes contain the column names and data types for the table. Each table has one or more column names that are underlined. These columns are the primary key for a table. This means that a row in that table can be uniquely identified if you know the values of each column in the primary key. For example, in the *REVIEW* table, the columns *AccountEmail* and *Sp_Id* are the primary key. If you know an account email and a service provider ID, you can uniquely identify no more than one row in the *REVIEW* table. Enforcing this constraint guarantees that each account can only write one review per business. In the *SERVICE_PROVIDER* table, the primary key is only the *Sp_Id* column. This means you can identify a service provider by its ID.

An arrow represents a foreign key constraint. A value in the column at the start of the arrow must exist in the column the arrow points to. For example, the CONTACT table has a column called *Sp_Id* that contains the service provider ID for a contact. It is a foreign key that references a column in the *SERVICE_PROVIDER* table which is also called *Sp_Id*. The values in the *CONTACT* table's column must exist in the *SERVICE_PROVIDER* table's column since every contact must be linked to a service provider by its ID.

There are only thirteen tables shown in Figure 5. Earlier, I said we would need a separate table for each service, and since there were twelve categories in the original directory, we would need twelve additional tables. However, Robert Willey thought there was a good chance that different data would need to be collected in the future. This would require new tables and columns to be created, and whenever the database structure changed, any applications connecting to the database would have to be updated. Since my group only had two semesters to work on the project, we would not be around to update the applications in the future. I thought of a solution that would allow the service tables to change, and the applications would automatically recognize the new columns in these tables. The structure of the service tables no longer needed to be defined in the diagram. In addition to this change, several new requirements were added as we designed the database.

# 4 Additional Requirements

## 4.1 Data Management

After the first milestone report, my group continued to meet with Willey. One of these meetings was scheduled to discuss the requirements and make sure the project was going in the right direction. Willey requested several changes to make it easier to manage the data. At this time, the requirements included a simple way to manage reviews on businesses. A moderator could flag a review if it was inappropriate. They could also view all flagged reviews and choose which ones to delete. If a user continued to write bad reviews, a moderator could suspend their account which would hide all reviews written by that account. Willey wanted to extend the idea of flagging and suspending to include service providers as well. He also wanted to be able to suspend individual reviews and flag accounts. I had to define what these terms meant and how they applied to service providers, reviews, and accounts. I also had to determine what actions could be performed on the data and who had permission to perform these actions.

## 4.2 Dynamic Service Tables

As I mentioned earlier, we designed the database so that different types of data can be collected, and applications connecting to the database automatically recognize this new data. This increased the complexity of both the database and any applications that needed to connect to the database. If the structure of the service tables was fixed, it would be easy for an application to query and update the data. A simple SQL query could be used to get the data since the table names and column names would be known. To update the data, a form could be designed that matched the format of the data.

However, since the structure of the service tables can change, an application would not know which tables and columns contain the data. It would also not know the proper format for the data. This information is called metadata which is data that describes other data. My solution

was to store all of this metadata in the database. When an application needs to look up the service data, it first looks up the metadata for a service table. This metadata includes the table name for the service, the names of each column in that table, and the proper format for the data. The application uses the table names and column names to get the data, and then it prints the data using the format information. If a new column or table needs to be added, its metadata is stored in the database so that the applications know about the new table or column.

To update the data, the application generates a custom form that allows data to be entered in the proper format. For example, the *Education* table may have a column called *InstrumentsTaught* which stores a list of instruments an instructor can teach. The instruments in this list are to be selected from a list of possible instruments which are also stored in the database. The application gets the table name, column name, data type, and a list of possible values that can be selected. The form is generated using this information. In this case, a check box can be displayed for each of the instruments. When the submit button is clicked, the selected values are stored in the database. What makes this difficult is that the form has to be generated rather than being fixed. If the *InstrumentsTaught* column was removed, the form would no longer show a data field for the column. Similarly, if a new column was added, a new data field would be shown in the form.

The end of the first semester was approaching, so we tried to finish all that we could before the break. Kevin and I wrote some documentation describing all the actions that could be performed with our system. Mark wrote SQL queries to create the database tables. For our second milestone report, we combined all the diagrams and documentation we had created. At the end of the semester, almost every requirement was documented and the database was created. However, we still had not written a single line of code. For the second semester, we moved on to application development.

## 5 Application Development

After the break, we started working right away. From our original requirements, we needed to create a web application and a mobile phone application. We decided to start with the web application. Mark set up version control on GitHub. Kevin and I started entering data into a test database. The group was productive for a week, but then progress slowed. After two weeks, we still made no progress on the web application. I knew someone would have to get the project started, so I created the home page and search page. Soon after, Kevin started working as well.

### 5.1 System Design

During the first semester, we discovered a problem that would affect the system design. The web host we were using had configured their databases so that connections could only be made from their server. This was probably done for security reasons, but we needed to connect to the database from a mobile application and from our own computers while we built the web application.

Several adjustments had to be made to work around this problem. First, while we were building the web application, we connected to a test database on our own computers rather than the database on the server. During deployment, we would have to change the connection strings so the database on the server could be used instead. We also had to create a web service so that the mobile application could use the database. The mobile application sends the proper parameters to the web service depending on what it needs to do. For example, it might need to look up some data or change something in the database. The web service checks these parameters to determine what to do and what data to return. Since the web service is located on the server, it is allowed to connect to the database and perform the requested action. We designed the web service and web application to use the same functions when connecting to the database. The system design is shown in Figure 6.

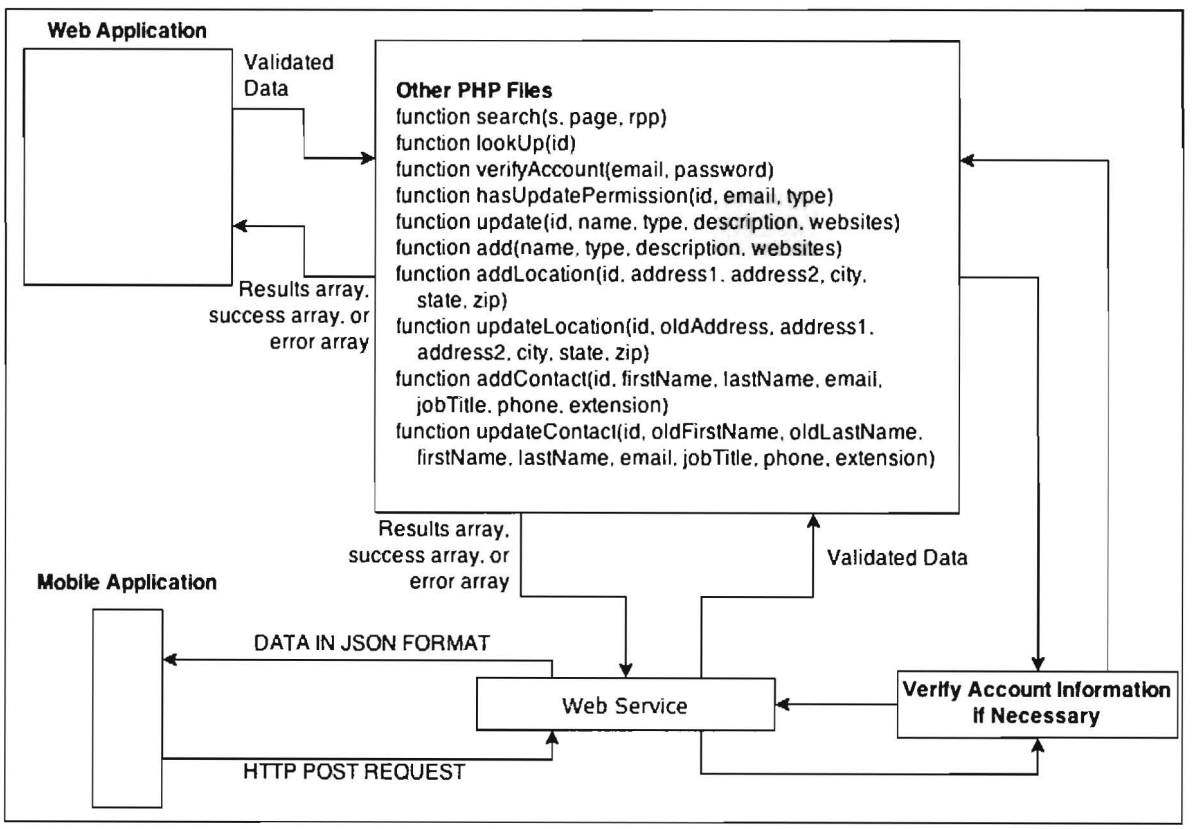


**Figure 6.** System Design

### 5.2 Programming and Testing

Kevin and I continued working on the web application. After finishing some of the basic features, I started working on the web service. Since we only had time to create one mobile application, we decided to create an iOS application which Willey preferred over an Android application. Mark was responsible for the iOS application since Kevin and I were working on the web application. By the time we finished writing our third milestone report, over half of the features in the web application had been implemented.

Whenever we finished a programming task, we would manually test the new features. We did this by entering test values into the forms we created and checking that the database was updated correctly. Security issues such as SQL injection and cross-site scripting were considered when testing. Also, we made sure that data could not be updated without authorization, and sensitive information such as emails and passwords was protected.

### 5.3 Deployment

Since we were behind schedule, I started working eight or more hours at a time on the project. Soon, the web application was ready to be uploaded to the server. After the files were uploaded, the database connection strings were changed so the application would connect to the database on the server. Every feature was tested again to make sure everything still worked. After we were satisfied with the amount of testing, Willey and his students started transferring the data from the old directory to the new database. We continued making small changes as needed. At the end of the semester, the web application was completed. Because of problems in our group, the iOS application development was postponed until the summer. However, Willey was satisfied with the web application. It turned out better than anyone could have expected.

## References

[1] Pressman, Roger S. *Software Engineering: A Practitioner's Approach*. 7th ed. New York: McGraw-Hill Higher Education, 2010.

[2] Elmasri, Ramez, and Shamkant B. Navathe. *Database Systems: Models, Languages, Design and Application Programming*. 6th ed. Boston: Pearson, 2011.